Hybrid Coevolutionary Algorithms vs. SVM Algorithms

Rui Li, Bir Bhanu and Krzysztof Krawiec
Center for Research in Intelligent Systems
University of California
Riverside, CA, 92521
{rli, bhanu}@vislab.ucr.edu, krawiec@cs.put.poznan.pl

ABSTRACT

As a learning method *support vector machine* is regarded as one of the best classifiers with a strong mathematical foundation. On the other hand, evolutionary computational technique is characterized as a soft computing learning method with its roots in the theory of evolution. During the past decade, SVM has been commonly used as a classifier for various applications. The evolutionary computation has also attracted a lot of attention in pattern recognition and has shown significant performance improvement on a variety of applications. However, there has been no comparison of the two methods. In this paper, first we propose an improvement of a coevolutionary computational classification algorithm, called Improved Coevolutionary Feature Synthesized EM (I-CFS-EM) algorithm. It is a hybrid of coevolutionary genetic programming and EM algorithm applied on partially labeled data. It requires less labeled data and it makes the test in a lower dimension, which speeds up the testing. Then, we provide a comprehensive comparison between SVM with different kernel functions and I-CFS-EM on several real datasets. This comparison shows that I-CFS-EM outperforms SVM in the sense of both the classification performance and the computational efficiency in the testing phase. We also give an intensive analysis of the pros and cons of both approaches.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology – *Classifier design and evaluation.*

General Terms

Algorithms.

Keywords

Machine learning, co-evolution, pattern classification, Support Vector Machines

1. INTRODUCTION

Support vector machines (SVM), introduced by Vapnik and coworkers in 1992 [1], has been noted as one of the best classifiers during the past 20 years. It is popular in bioinformatics, text analysis and pattern classification [2]. Table I lists the pros and cons of SVM. It is not a perfect classifier.

Table I. Pros and cons of SVM.

• Pros	• Cons
• It is independent of the feature dimensionality thus immune from the "curse of dimensionality". • With the kernel tricks, different discriminant functions can be obtained by using different kernel functions. • The boundary is determined only by its support vectors, namely SVM is robust against changes of all vectors but its support vectors. • Different SVM classifiers constructed by using different kernels (polynomial, RBF, linear) extract almost the same support vectors.	• As a supervised classifier, it needs large amount of labeled training data. • The best choice of the kernel function for a given problem is still a research problem. • The testing speed depends on the number of support vectors, so it could be slow. • The optimal design for multi-class SVM classifiers is also a research area. • It has convergence problem with large datasets.

On the other hand, during the past decade, another methodology called *Co-Evolutionary Computation* (CEC) has attracted lots of attention in pattern recognition and machine learning. It has shown great performance on image representation, object recognition and image retrieval, etc. As a classifier, it is comparable to SVM for classification performance [3].

However, there has been no shoulder-to-shoulder comparison between CEC and SVM yet. So in this paper, first, we select one of the recent CEC based classification algorithm *Coevolutionary Feature Synthesized EM* (CFS-EM) [3], make improvements to it and call it I-CFS-EM. Then in the experiments, we compare it with SVM using different kernels on three image datasets. The experimental results show that I-CFS-EM has higher classification accuracy and better convergence than CFS-EM and it outperforms SVM in both the sense of classification performance and testing efficiency. We also discuss about the pros and cons of both I-CFS-EM and SVM and their suitability for different applications.

The rest of this paper is organized as follows. Section 2 gives some related work. Section 3 describes the I-CFS-EM algorithm in detail. Section 4 presents extensive comparisons between I-CFS-EM and SVM. Finally, section 5 concludes the paper.

2. RELATED WORK

The handling of high feature dimensionality and the labeling of training data are the two major challenges in pattern recognition. To handle the high feature dimensionality, there are two major approaches. One is to use special classifiers which are not sensitive to dimensionality, for example, SVM algorithm and boosting algorithm. The other way is feature selection. It is defined as selecting a subset of features or constructing new features that are useful to build a good predictor for classification.

A variety of interesting feature selection approaches has been proposed. Lee and Seung [4] use a *non-negative matrix factorization* (NMF) approach that yields a part-based representation instead of a holistic representation. It is found to be useful for face recognition [5] and document retrieval [6] . He et al. [7, 8] propose the approach of *locality preserving projections* (LPP) for face analysis and image retrieval. It preserves local information by detecting a nonlinear manifold structure. Genetic algorithms and genetic programming have also been used for feature selection for object recognition [9] and image retrieval [10]. All the above approaches are supervised classifiers. So they need a large amount of labeled data to train large datasets. But manual labeling is expensive and subjective.

On the other hand, unsupervised learning, for example, the *Expectation Maximization* (EM) algorithm [11] which is commonly used does not have the labeling problem, but it has two limitations. 1) "The curse of dimensionality," in high dimensional feature space it needs a large amount of data and the computational cost varies exponentially with the dimensionality, which is not always available. 2) As a hill-climbing kind of method, the convergence of EM algorithm is guaranteed only at a local maximum.

Thus neither supervised learning nor unsupervised learning can solve both the high dimensionality and the labeling problem simultaneously. Therefore, the learning usually can only be carried out with the hybrid of both labeled and unlabeled data, called *transductive learning* [12].

The basic idea of *transductive learning* is that it combines both labeled and unlabeled data in training. In this scheme, labeled data provide the initialization and validation of the classifier, and the unlabeled data captures the statistical characteristics of the dataset and boost the classifier. Various transductive learning methods have emerged during the past 10 years. Joachimes [13] uses SVM scheme to maximize the margin for both the unlabeled data and the labeled data by assigning the unlabeled data to proper classes. Bargeron et al. [14] propose a boosting-based transductive learning. The idea is to construct a diversity of unsupervised models of unlabeled data using a clustering algorithm. These models are then exploited to construct a number of hypotheses using the labeled data and the learner selects a hypothesis that minimizes a transductive error bound. Wu et al. use a linear feature transformation (MDA) + EM scheme to solve the transductive learning problem [15, 16]. Li et al. [3] propose another transductive learning scheme, a hybrid of non-linear feature synthesis and EM algorithm, called CFS-EM.

Transductive learning has been applied in image retrieval [3, 16], color segmentation [15], text classification [13], text detection [14] and digit recognition [17].

As the latest member of transductive learning family, CFS-EM has shown a better performance than D-EM and comparable performance with SVM [3].

In this paper, we build a new classifier based on CFS-EM, called the *Improved CFS-EM* (I-CFS-EM). Besides of keeping all the advantages of CFS-EM, it improves CFS-EM on the training speed, convergence and classification performance. In this paper we do comprehensive comparison between I-CFS-EM and SVM on real image datasets. We also discuss the pros and cons of each method, and their suitable applications.

3. I-CFS-EM ALGORITHM

To help the readers understand the paper, we list the definition of all the key symbols in Table II.

Table II. Definition of the key symbols used in this paper.

C	The number of classes
D	The original feature dimensionality
d	The reduced feature dimensionality
L	The labeled training data set
U	The unlabeled training data set
T	The total number of training data ($L+U$)
l	L in low dimension
u	U in low dimension
r	The percentage of L in T, $r = L/(L+U)$
$\mathcal{X}$	The dataset in EM algorithm
$\mathcal{Y}$	The labels for $\mathcal{X}$ in EM algorithm and the labels for $L+U$ in I-CFS-EM algorithm
W	L, $U + \mathcal{Y}$.
λ	The percentage of support vectors in L
β_i,γ,τ,t	The parameters for SVM classifier
CO(.)	Composite operator vector
POP	The whole population in I-CFS-EM
K	The maximum composite operator size
$\alpha_i,\mathbf{u}_i,\mathbf{\Sigma}_i$	Parameters of Gaussian mixture (component weights, means and covariance matrices)
θ	The whole parameter set for Gaussian mixture ($\alpha_i+\mathbf{u}_i+\mathbf{\Sigma}_i$ in EM algorithm) or the Bayesian classifier in I-CFS-EM.

I-CFS-EM is a combination of *coevolutionary feature synthesis* (CFS) and EM algorithm. So we discuss them separately before the whole algorithm.

3.1 Coevolutionary feature synthesis

Coevolutionary feature synthesis is implemented with a *coevolutionary genetic programming* (CGP) approach [9]. It creates low dimensional synthesized feature vectors from the original high dimensional feature vectors. The synthesized features are obtained by applying a series of operators (called a *composite operator vector*) to the original features. These operators are binary trees with simple operations, called *primitive operators* (12 simple operators: ADD, SUB, MUL, DIV, ADDC, SUBC, MULC, DIVC, LOG, SQRT, MIN2, MAX2) as the inner nodes and original features as the leaf nodes. The fitness function is the classification accuracy of the Bayesian classifier in the low dimensional synthesized feature space. This helps the objects in the same class to form a Gaussian component no matter how they are distributed in the original space. Actually, with the help of the fitness function, we force the distribution in the low dimension to be a *Gaussian Mixture Model* (GMM) even though the original features are not GMM and the operators are non-linear. In another word, we use the GMM as a target not an assumption because the EM step in the loop is based on the GMM assumption. We follow the CGP algorithm proposed in [9] to find the "best" composite operator vector. During the training phase, CGP runs on the labeled training data and evolves to get the best composite operator vector and a Bayesian classifier in the synthesized feature space. The during the testing phase, a composite feature vector (low dimensional vector) is firstly obtained by applying the composite operator vector on the original features of the testing sample, and then the Bayesian classifier is used for its classification.

3.1.1 Generational Coevolutionary Genetic Programming (CGP)

Generational coevolutionary genetic programming is shown as in Algorithm 1. This algorithm evolves composite operators. The composite operators (binary trees) in the initial subpopulations are randomly generated. A composite operator is generated in two steps. In the first step, the internal nodes of the tree representing the composite operator is randomly determined in a recursive manner as long as this number is smaller than a given number. In the second step, after all the internal nodes are generated, the original features are randomly picked and attached to the leaf nodes. The GP operations are applied in the order of crossover, mutation and selection. In addition, an elitism replacement method is adopted to keep the best composite operator from generation to generation. To evaluate the *j*th composite operator CO_j in step 5) in Algorithm 2, the current best composite operator in each of the other subpopulations are selected and combined with CO_j to form a composite operator vector where composite operator from the *j*th subpopulation occupies the *j*th position in the vector ($j= 1,\dots, S$). The composite operator vector is run on the original features of the training images to get composite feature vectors and they are used to build a Bayesian classifier. The classification accuracy of the Bayesian classifier is the fitness of the composite operator vector and CO_j.

Algorithm 1: Generational Coevolutionary genetic programming (CGP)

```
Input: primitive feature vectors, class number, number of subpopulations
Output: composite operator vector, Bayesian classifier in low dimension.
Begin:
1. Randomly generate $S$ subpopulations of size $M$ and evaluate each
   composite operator in each subpopulation individually.
2.  FOR gen = 1 to G DO
      FOR i =1 to S DO
        1) Keep the best composite operator in subpopulation $P_i$ .
        2) Perform crossover on the composite operators in $P_i$ until the
           crossover rate is satisfied and keep all the offspring from
           crossover.
        3) Perform mutation on the composite operators in $P_i$ and the
           offspring from crossover with the probability of mutation rate.
        4) Perform selection on $P_i$ to select some composite operators and
           combine them with the composite operators from crossover and
           mutation to get a new subpopulation $P_i'$ of the same size as $P_i$.
        5) Evaluate each composite operator $CO_j$ , $j = 1, \dots, M$ in $P_i'$.
        6) Perform elitism replacement.
        7) Form the current best composite operator vector consisting of
           the best composite operators from corresponding
           subpopulations and evaluate it. If its fitness is above the fitness
           threshold, goto 3.
      END FOR
    END FOR
3. Select the best composite operator from each subpopulation to form the
   learned composite operator vector and output it.
End
```

3.1.2 Parameters

The key parameters are the number of sub-population S, the size of sub-population M, the number of generations G, the crossover and mutation rates, the tournament size and the fitness threshold. CGP stops whenever it finishes the specified number of generations or the performance of the Bayesian classifier is above the fitness threshold. After the termination, CGP selects the best composite operator of each sub-population to form the learned composite operator vector and the Bayesian classifier to be used during the testing. The number of sub-population and the size of sub-population are dataset dependent parameters and they are related with the size of the search space. Normally they are a very small percentage of the size of the search space. We have determined these two parameters and the number of generations based on empirical experiments from our previous work [3, 10]. Other parameters (crossover rate, mutation rate and tournament size) are well explored in the literature on evolutionary computation, so they are chosen based on the experience [18, 19].

3.2 EM Algorithm

We assume feature vectors in the low dimension follow a C component *Gaussian Mixture Model* (GMM). They can be regarded as samples of a d-dimensional random variable X, where x = $[x_1, \dots x_d]^T$ represents a particular sample. Its probability density function is defined as $p(\mathbf{X}\mid\theta) = \sum_i^C \alpha_i f_i(\mathbf{X}\mid\theta_i) = \sum_i^C \alpha_i f_i(\mathbf{X}\mid\mathbf{u}_i,\Sigma_i)$. In this definition, $\alpha_1, \dots \alpha_C$ are the *mixture probabilities*, so they must be positive and sum up to 1. θ_i is the set of parameters for the *i*th Gaussian component, which includes mean $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$. Thus, $\theta \equiv \{\alpha_1, \dots \alpha_C, \theta_1, \dots \theta_C\}$ is the complete set of parameters needed to specify the mixture. Given a set of N independent samples of $\mathbf{X}$: $\mathcal{X}=\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$, the goal is to find θ which maximizes log p($\mathcal{X}|\theta$) (maximum likelihood). *Expectation-Maximization* (EM) is such an algorithm [20] to find θ. It is based on the interpretation of $\mathcal{X}$ as incomplete data. The missing part is a set of N labels $\mathcal{Y}=[\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)}]$ associated with the N samples, indicating which component produced each sample. $\mathbf{y}^{(n)}=[y_1^{(n)},\dots, y_C^{(n)}]$, where $y_k^{(n)}=1$ and $y_j^{(n)}=0$ for $j \neq k$, means that sample $\mathbf{y}^{(n)}$ was produced by the *k*th component. The complete log-likelihood is

$$\log p(\mathcal{X},\mathcal{Y}|\theta) = \sum_{n=1}^{N}\sum_{i=1}^{C} y_i^{(n)} \log\left[\alpha_i p\left(\mathbf{x}^{(n)}\middle|\theta_i\right)\right] \qquad (1)$$

EM algorithm produces a sequence of estimates $\{\hat{\theta}(t)$, $t = 0, 1, 2,\dots\}$ by alternatively applying the following E-step and M-step until some convergence criterion is met.

- **E-step** computes the conditional expectation of the complete log-likelihood, given $\mathcal{X}$ and the current estimate $\hat{\theta}(t)$. The result is the so-called *Q*-function shown in equation (2). In this equation, because of the linearity of log $p(\mathcal{X},\mathcal{Y}|\theta)$ with respect to $\mathcal{Y}$, we only need to compute the conditional expectation $\mathcal{Z} \equiv E[\mathcal{Y}|\mathcal{X}, \hat{\theta}(t)]$. It is explicitly given by equation (3), where $z_i^{(n)}$ is a posteriori probability that $y_i^{(n)}=1$, after observing $\mathbf{x}^{(n)}$.

$$Q(\theta,\hat{\theta}(t)) \equiv E[\log p(\mathcal{X},\mathcal{Y}\mid\theta)\mid\mathcal{X},\hat{\theta}(t)] = \log p(\mathcal{X},\mathcal{Z}|\theta) \qquad (2)$$

$$z_i^{(n)} \equiv E[y_i^{(n)}\mid\mathcal{X},\hat{\theta}(t)] = \Pr[y_i^{(n)}=1\mid\mathbf{x}^{(n)},\hat{\theta}(t)] = \frac{\hat{\alpha}_i(t)p(\mathbf{x}^{(n)}\mid\hat{\theta}_i(t))}{\sum_{j=1}^{C}\hat{\alpha}_j(t)p(\mathbf{x}^{(n)}\mid\hat{\theta}_j(t))} \qquad (3)$$

- **M-step** updates the parameter estimation to maximize the *Q*-function.

3.3 I-CFS-EM Algorithm

The I-CFS-EM algorithm is described in Algorithm 2. In the initialization, CGP is applied on the labeled training data (L) to get a composite operator vector CO(·) and a Bayesian classifier represented by the Gaussian distribution parameter set θ. The population corresponding to the best composite operator vector is saved (POP). Both the labeled training data (L) and the unlabeled training data (U) are transformed into the low dimension (l and u) using this composite operator vector CO(·). In I-CFS-EM iteration, firstly the *'EM hill climbing'* step is applied on this low dimension dataset ($l+u$) to find a locally optimal Bayesian classifier. EM stops when the Bayesian parameters (θ) do not change for two consecutive iterations. At this time, both L and U are "labeled" by the Bayesian classifier. Labels of L will be changed to their ground-truth labels (step 2). In the *'Jump out of local maximum'* step, CGP is applied on the "labeled" whole dataset to find a better composite operator vector (step 6) and to update the Bayesian classifier. The stored population is used as the initial population for CGP and it is updated by the new population to keep the performance increasing. The *'Hill climbing'* and *'Jump out of local maximum'* steps iterate until one of the three termination conditions is reached: 1) a certain number of iterations is run, 2) a satisfactory classification performance is reached, or 3) the fitness value does not change for 5 iterations.

Algorithm 2: I-CFS-EM

```
Input:  labeled training dataset L from C classes in the original feature
        space
        Unlabeled training dataset U in the original feature space
        Synthesized feature dimension d
        CGP parameters (sub-population size, crossover rate, mutation
        rate, maximum composite operator size, fitness value,
        tournament size and generation number)
Output: Composite operator vector CO(•)
        Bayesian classifier in the low dimension θ
Begin:
   Initialization:
      [CO(•), θ] = CGP(L). (randomly initialized)
      Save the population corresponding to the best CO(•) as POP.
      l = CO(L), u = CO(U).
   I-CFS-EM iteration:
    Hill climbing:
      1) Get labels (Y) for l + u (by calculating Z based on equation (3)).
      2) Change the labels (Y) of l to the ground-truth labels.
      3) Update θ based on Y.
      4) If θ does not change much, goto 5), otherwise, goto 1).
    Jump out of local maximum:
      5) W = L,U + Y. // W is the "labeled" whole training dataset (L, U
         together with their labels Y)
      6) [CO(•),θ] = CGP(W) (using POP as the initial population
         and updating POP at the end).
      7) l = CO(L), u = CO(U), goto 1).
End
```

In the hybrid approach, CFS-EM initializes the entire population randomly at each iteration, which gives up all the training work from the previous iteration. *Different with CFS-EM, the hybrid I-CFS-EM saves the population at the CGP step of each iteration and uses it as the initial population for the CGP step in the next iteration* (as highlighted in Algorithm 2 with bold font). This helps to obtain a better convergence and a better classification performance. *I-CFS-EM also defines three new termination conditions to speed up the training.* In addition, *Exploration and exploitation scheme is used within CGP to decrease the mutation rate from iteration to iteration*, such that population with big variance could be explored using high mutation rate in the early iterations and the convergence is assured with the low mutation rate in the later iterations. The mutation rate is fixed for all the generations in one iteration. The advantages of I-CFS-EM are: 1) It synthesizes low-dimensional features based on the CGP algorithm, which yields near "optimal" nonlinear transform; 2) The unlabeled data can be boosted with the help of the class distribution learning using the CGP feature synthesis approach; 3) The possibility exists of helping EM to jump out of local maximum and reach the global maximum.

Table III compares I-CFS-EM with SVM in both the classification capability and computational complexity.

Table III. Comparison between I-CFS-EM and SVM in theoretical considerations.

	I-CFS-EM	SVM
Feature dimensionality change	High → low	High → higher
Problem of kernel selection	No	Yes
Sensibility to parameter selection	No	Yes
Meaningful model for the classifier	Yes	No
Multi-class handling	Natural	Non-natural
Stability	Medium	High
Training time	Long	Short
Testing time	Short	Long

In the following we explain Table III in detail. 1) Both of them synthesize new features from the original data. The difference is that I-CFS-EM transforms the original feature vectors into a low dimension while SVM projects them into a higher dimension. Both of the transformed features have no physical meaning. 2) The performance of SVM highly depends on the selection of kernel function. Different kernel functions give quite different performance. Some kernel functions (e.g. sigmoid) do not even converge on some dataset. I-CFS-EM has no such constraint. 3) Even a good kernel function is picked for SVM, the parameter setting has a big impact on the performance, so cross validation is required to choose the best set of parameters. On the other hand, even though I-CFS-EM has a lot more parameters (population size, crossover/mutation rate, etc.) to set, the evolutionary computation community has done a lot of research on it and the effect of parameters is reasonably well understood. So the default parameters generally do a good job. 4) I-CFS-EM classifier is obtained in the form of a feature transformation and a Bayesian classifier. However, SVM generates black box models in the sense that it has no ability to explain, in an understandable form. 5) Given in a form of a Bayesian classifier in the low dimension, I-CFS-EM is born for multi-class tasks. SVM is designed for two class problems. Thus it can only use one-against-one (or one-against-the-rest) + voting mode to handle multi-class problems, a non-natural way. So it is not as efficient as I-CFS-EM for testing. 6) As a learning approach, I-CFS-EM gives different classifiers at each run. Some time it overfits, but it can be avoided by integrating a criterion like *Minimum Description Length* in the fitness function. SVM is less likely to overfit and gives the same result all the time. So I-CFS-EM is not as stable as SVM. 7) From the computational cost perspective, I-CFS-EM is slower than SVM in the training phase but much faster than SVM in the testing phase. The long training time is due to the exploitation among the large population. The short testing time is because that the feature transformation enables the test to be done in the synthesized low-dimensional space, while SVM has to make the classification computation in the original high dimensional space using all the support vectors. This makes I-CFS-EM suitable for time critical applications even though it requires long training time. There is no free lunch, right?

4. EXPERIMENTAL RESULTS

To evaluate the efficacy of I-CFS-EM approach for classification, we apply it on three real image datasets of small, medium and large sizes. We compare it with RBF-SVM, Linear-SVM and Poly-SVM (corresponding to radial basis function, linear and polynomial kernels) on both the classification performance and the test efficiency. We use image datasets because for image retrieval, the labeling is a problem and the query efficiency is a key concern.

4.1 Datasets

- ***Corel-1500*:** We select 1200 images belonging to 12 classes from Corel Stock photo library (Mayan & Aztec Ruins, horses, owls, sunrises & sunsets, North American wildflowers, ski scenes, coasts, auto racing, firework photography, divers & diving, land of the Pyramids and lions) because they have high semantic consistency for each class and it has been used in some previous work [21] for image retrieval evaluation. The images in each class have similar visual features. We assume that the images in each class in the feature space forms a Gaussian cluster in the feature space. Then we add 300 images from three other CDs in the library (hawks and falcons, tigers and tulips) to form *Corel-1500*. The three new CDs are merged to owl, lions and North American wildflowers, respectively. Thus, *Corel-1500* still has 12 classes. The purpose of *Corel-1500* is to demonstrate that CGP can map the original features to a low dimensional space and make their distribution a Gaussian no matter how the original features are distributed. Since each class is no longer a Gaussian cluster in the original feature space, this database is challenging for pure EM and linear transformation D-EM [16].

- ***Corel-6600*:** This dataset is also extracted from Corel stock. It contains 50 classes and there are ~100 images in each class, which give a total of 6600 images. This dataset is to test the scalability of the algorithm.

- ***Corel-10038*:** This dataset is also obtained from Corel stock. It contains 56 classes and a total of 10038 images. The same dataset was used by Yin el al. [22] for relevance feedback experiments. The dataset is quite unbalanced for different classes. The class size varies from 695 images to 20 images. The small classes do not have enough samples to fit the Gaussian distributions. So we redistribute samples from the three smallest classes (obelisk, golf course and national flag) into the visually closest classes. Thus this dataset has 53 classes and the minimum class has 60 images. All the other classes have varying number of images: 62, …, 395,…, 695. This dataset can test the ability to handle unbalanced classes. This dataset is larger than *Corel-6600* and the classes are unbalanced. Thus it can test the algorithm's scalability in a more comprehensive way than *Corel-6600*.

For *Corel-1500* and *Corel-6600*, each image is represented by 16 texture features, 6 color features and 18 structural features, a total of 40 features. For *Corel-10038*, only texture features and color features are used (totally 22). The 16 texture features are means and standard deviations derived from 8 Gabor filters (2 scales and 4 orientations). The 6 color features are means and standard deviations of the HSV color channels. The 18 structural features are extracted using the water-filling approach [23].

4.2 Experimental settings

All the features in the above three datasets are normalized (to range [0,1]). The whole dataset is split into halves, one half for training and the other half for testing. A percentage of the training data (20%, 40%, 60% and 80%) is defined as the labeled training data (L) and the rest as the unlabeled training data (U). For a certain percentage of labeled data, the average classification accuracies, the precision-recall curves and the query computational cost on the testing data are compared among the four classifiers. I-CFS-EM reduces the feature dimensionality from 40 to 6 for *Corel-1500 and Corel-6600*, and from 22 to 4 for *Corel-10038*. Because of the unbalanced classes in *Corel-10038* even after redistributing the samples from the smallest classes, some small classes (minimum class size is 6 for the training data when 20% of training data are labeled) do not have enough samples to fit a 6 dimensional Gaussian distribution. So for this dataset, we decrease the features dimensionality to 4.

We use LIBSVM [24] to evaluate the performance of SVM. We use RBF, linear and polynomial kernel functions. We do not use the sigmoid function because kernel matrix using sigmoid may not be positive definite. Actually, we have tried sigmoid kernel in the experiment and the result is singular. To handle multi-class problem, it works in a one-against-one + voting mode. We do a five-fold cross validation to obtain the best set of parameters for each kernel function.

In addition, we also try *Transductive SVM* (TSVM) [13] on the datasets. We use SVM^{light} [25] to evaluate its performance. To handle the multi-class problem, it works in a one-against-the-rest + voting mode. The accuracy is very low so we do not show it here. This is because it is designed for binary classification tasks. The one-against-the-rest mode makes the two classes too unbalanced and TSVM cannot find the good margin for them.

For I-CFS-EM, we run the program for 10 times. The maximum vote of the classifications of each test image from the 10 runs is used as the final classification of that image. The I-CFS-EM parameters are: (a) number of sub-population: 6 for *Corel-1500 and Corel 6600*, 4 for *Corel-10038*; (b) sub-population size: 100; (c) number of generation: 100; (d) crossover rate: 0.6; (e) mutation rate: *exploration and exploitation* scheme is used to decrease the mutation rate from 0.1 to 0.01 with step size 0.01 in 10 iterations; (f) tournament size: 5; (g) fitness threshold: 1.0.

4.3 Classification performance comparisons

Figure 1 shows the classification accuracy comparison of the four classifiers on the three datasets. The accuracy for all the four classifiers increases as the percentage of labeled data increases. As a semi-supervised learning method, I-CFS-EM has higher accuracy than the supervised learning method SVM (with all three kernel functions). We can also see big gaps among the performance of the three kernel functions. RBF is the best as recommended by the literature. Polynomial kernel has the worst performance. In our previous work [3], RBF-SVM is not as good as Linear-SVM because we used the default SVM parameters. In the new experiments, after applying cross validation, we get fair evaluation of SVM and RBF-SVM outperforms Linear-SVM. In Figure 1 (a), result of *Corel-1500*, accuracy curve of CFS-EM is added. We could also see the obvious accuracy improvement from CFS-EM to I-CFS-EM, which is 10% ~ 15%. Figure 2 shows the *precision-recall* (PR) curves of the four classifiers on the three datasets when 20% and 60% of the training data are labeled. Compared with the accuracy curves, a PR curve gives us more detailed information about the performance. In these plots, we can see the same trend as in the accuracy curves: I-CFS-EM > RBF-

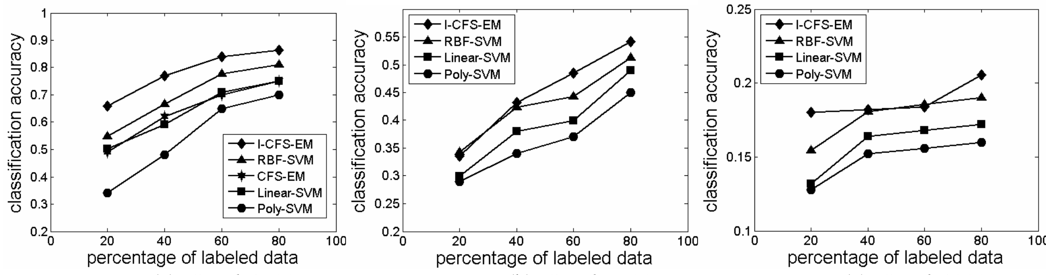
SVM > Linear-SVM > Poly-SVM; as the percentage of *L* increases, the overall performance of all the four classifiers increase; the performance of I-CFS-EM is higher than SVM, and the gap increase as the number of classes increases.

Table IV. 95% Confidence intervals of I-CFS-EM.

	r = 20%	*r* = 40%	*r* = 60%	*r* = 80%
Corel-1500	0.66±0.038	0.77±0.034	0.84±0.029	0.863±0.027
Corel-6600	0.34±0.016	0.432±0.017	0.485±0.017	0.54±0.017
Corel-10038	0.18±0.0106	0.182±0.011	0.184/±0.01	0.206±0.011

Table IV shows the 95% confidence intervals of I-CFS-EM on the three datasets. We could see the confidence is getting higher when dataset size increase.

Figure 3 shows the 2D projections of the low dimension features (6D) of *Corel-1500*. Since there is not a good way to present the 6D feature vectors on the 2D paper, they are projected to 2D. Every sub-figure shows the projection of the samples from two classes (blue circles and purple crosses) into two dimensions where they have the best separation. The ellipses are the Gaussian ellipses with two times the standard deviation of each class. Because of the lack of space, we only show some representative combinations. Each class in *Corel-1500* is approximately a Gaussian distribution even though their original distribution is not, which justifies the advantage of non-linear transformation CGP over linear transformation *Multi Discriminant Analysis* [16].



(a) *Corel-1500* (b) *Corel-6600* (c) *Corel-10038*

Figure 1. Classification accuracy comparison at different percentage of labeled data (*r*) for the three datasets. Accuracy order: I-CFS-EM > RBF-SVM > Linear-SVM > Poly-SVM. All accuracies increase as *r* increases.

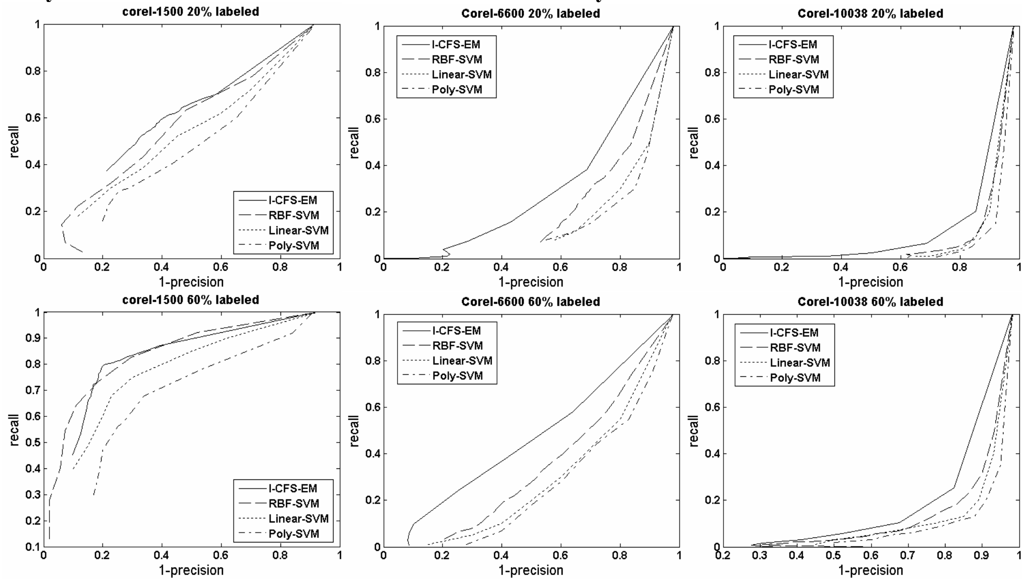


Figure 2. Precision-Recall curves of I-CFS-EM and SVM (different kernels) at different *r* (20% and 60%) for the three datasets.

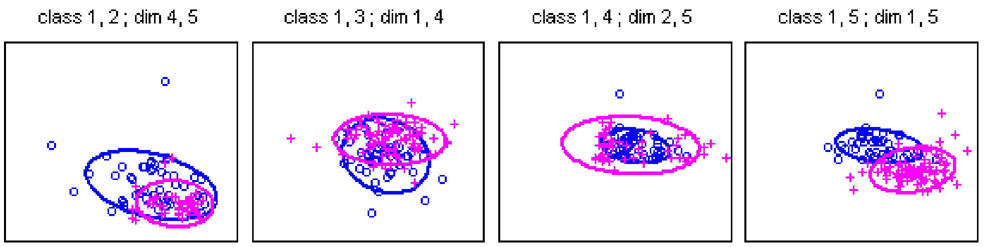


Figure 3. Projection of synthesized feature vectors of *Corel-1500* in 2D (the two axis are the two projection dimensions which give the best separation for the given two classes). Each class forms an ellipse, which satisfies the Gaussian mixture assumption.

4.4 Comparison of computational cost in the testing phase

For time critical applications, testing efficiency is a big concern. In this section, we will show that I-CFS-EM is more efficient than SVM in testing, both theoretically and experimentally, especially when the original feature dimensionality is high. The parameters that affect query time for I-CFS-EM and SVM are: class number *C*, original feature dimensionality *D*, reduced feature dimensionality *d*, the amount of training data *T*, the percentage of labeled training data *r* and maximum composite size *K*.

For I-CFS-EM, the maximum computational cost of one query is given by equation (4), in clock cycles (shortened as cc). The first term is for the composite feature transformation. Given *K* as the maximum number of nodes in a tree, the maximum number of primitive operators used is *K* when all the primitive operators have only one operand and all the *K* nodes are used. Assume the most complex computation SQRT (square root) (as mentioned in section 3.1) is used at each node, which costs 22.3 cc[1], then *d* composite operators need a maximum of 22.3*Kd* cc. The second term is the calculation of the Gaussian probability to *C* classes. To find the nearest class we need to find the largest probability among *C* classes, which needs *C* cc, which gives the third term.

$$22.3Kd + C(d^2 + 26.5) + C \tag{4}$$

The classification of x between two classes for RBF-SVM, Linear-SVM and Poly-SVM is defined by equation (5)~(7), where, x_i, $i = 1,\dots, nSV$ are the support vectors from the two classes. So $nSV = 2\lambda rT/C$. The norm calculation costs *D* cc and the exponential costs 26.5 cc for RBF-SVM. The inner product calculation costs *D* cc for Linear-SVM. The power calculation costs *Dd* cc for Poly-SVM. In addition, multi-class SVM is carried by one-against-one + voting mode, so the total computational cost is giving by equation (8)~(10).

$$f(x) = \operatorname{sgn}\left\{\sum_{i=1}^{nSV} \beta_i \exp\left(-\gamma \|x_i - x\|^2\right) + t\right\} \tag{5}$$

$$f(x) = \operatorname{sgn}\left\{\sum_{i=1}^{nSV} \beta_i \left(x_i^T x\right) + t\right\} \tag{6}$$

$$f(x) = \operatorname{sgn}\left\{\sum_{i=1}^{nSV} \beta_i \left(\gamma\left(x_i^T x\right) + \tau\right)^d + t\right\} \tag{7}$$

$$C(C-1)/2 \cdot 2\lambda rT/C \cdot (D + 26.5) + C \tag{8}$$

$$C(C-1)/2 \cdot 2\lambda rT/C \cdot D + C \tag{9}$$

$$C(C-1)/2 \cdot 2\lambda rT/C \cdot (Dd) + C \tag{10}$$

The percentage of support vectors λ is decided by the class number *C*, feature dimensionality *D* and percentage of labeled training data *r*. From the real data in our experiments, the percentage of supported vectors is close for different kernel functions, as mentioned in the introduction (Table I. Pros and cons of SVM.). We use *Least Mean Square Error* (LMSE) estimation to get a linear combination of *C, D* and *r* as the estimation of λ. The fitting result is shown in equation (11). The small MSE validates the linear assumption.

$$\lambda = 0.004645*C + 0.002046*D - 0.1532*r + 0.70624, \quad MSE = 0.0011 \tag{11}$$

Figure 4 shows the computational cost comparison between I-CFS-EM and SVM with reference to the percentage of labeled training data for *Corel-1500*. The solid line is the maximum cost of I-CFS-EM and the other three lines are the cost of SVM with different kernels. It can be seen that they all increase linearly with the percentage of labeled training data while the cost of I-CFS-EM does not. Linear-SVM has the lowest slope and the Poly-SVM has the highest slope. This result is reasonable because from the definition of the kernel functions, they have different computational complexities. The figure shows that even when only 3% of the labeled training data are used, SVM needs more computation than I-CFS-EM in the testing phase.

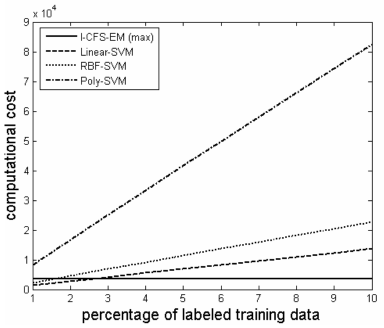


Figure 4. Computational cost: I-CFS-EM vs. SVM (For *Corel-1500*, *C* = 12, *D* = 40, *d* = 6, *T* = 749, *r* = 1%~10%, *K* = 20).

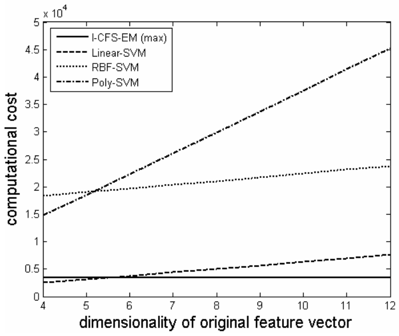


Figure 5. Computational cost: I-CFS-EM vs. SVM (For *Corel-1500*, *C* = 12, *D* = 4~12, *d* = 6, *T* = 749, *r* = 20%, *K* = 20).

From Figure 5 we can see that when the percentage of labeled data is fixed, computational cost of SVM increases as the number of dimensions of the original feature vector increases while it is fixed for I-CFS-EM. Even when the dimension of original feature vector is only 6, Linear-SVM surpasses I-CFS-EM. The reason that I-CFS-EM is computationally more efficient than SVM in the query phase is that I-CFS-EM testing works in a low dimensional synthesized feature space while SVM works in the original high dimensional feature space using lots of the training samples. In the experiments on *Corel-1500*, the total testing time for 749 images is ~5 seconds for Linear-SVM while it is less than 1 second for I-CFS-EM.

In summary, I-CFS-EM is more efficient than SVM in the testing phase, which makes it more suitable for time critical applications.

[1] The computational cost (cc) of the operations used in our calculation (SparcStationII) *(http://www.mathematik.uni-kl.de/~zca/Reports_on_ca/11/paper_html/paper.html)*

add	1	sub	1	mult	1.1	div	3.1
sqrt	22.3	log	20.3	exp	26.5	comp	1

4.5 Discussion

Based on the above comparison, we can see: 1) the improvement of I-CFS-EM over CFS-EM is big; 2) I-CFS-EM has a higher classification performance than SVM; 3) the classification performance of SVM is highly kernel dependent; and 4) I-CFS-EM is much faster than SVM in the testing phase, and the time does not change with the original feature dimensionality or the percentage of labeled data.

However, I-CFS-EM is not flawless. Its training time is longer compared with SVM. For some classes with too few samples, it cannot converge because the Gaussian model for each class in the low dimension requires enough samples to make the covariance matrix non-singular. As a randomly generated classifier, only the average performance is guaranteed. More than one run may be required to achieve a good performance. In addition, as a transductive learning method, our approach obeys the basic assumption that the unlabeled data has a similar distribution with the labeled data. When this assumption is violated, the performance will degrade. To handle the situation when the real number of components is different with the given component number, we can add the component estimation method into the iteration. In addition, the predicted labels on the unlabeled patterns have the same weight as the given labels on the labeled patterns. Thus, the mis-predicted labels may dominate in the parameter estimation process of EM, particularly when there are a lot of unlabeled data. So the use of different weights can possibly make CGP and EM cooperate better and improve the classification accuracy.

In general, there is no perfect classifier. They have their own suitable applications. Between I-CFS-EM and SVM, if the classes are uneven and the training time is a concern, e.g., online text classification for search engine design, SVM with RBF kernel will commonly give an acceptable performance; however, if the testing efficiency are key concern, e.g., image retrieval, I-CFS-EM will be a better choice.

5. CONCLUSIONS

Handling high dimensional feature vectors and labeling of training data have been two challenging problems in pattern classification. To solve these problems, we propose an improvement of a CGP/EM hybrid algorithm, called I-CFS-EM algorithm. It combines a transductive learning scheme with a feature dimensionality reduction technique. In this algorithm, CGP is used to synthesize lower dimensional feature vectors from the original high dimensional feature vectors so that EM can be used to form a Gaussian mixture with limited data. Labeled training data is used to initialize the synthesized feature transformation, and unlabeled training data is used to boost the learned classifier. We apply I-CFS-EM on small, medium and large datasets, evaluate and compare it with SVM and find it outperforms SVM in both classification performance and testing efficiency. As discussed in section 4.5, both SVM and I-CFS-EM have limitations. So proper classifier should be picked based on the requirements of the application.

6. REFERENCES

[1] B. E. Boser, et al., "A training algorithm for optimal margin classifiers," in *5th Annual ACM Workshop on COLT*, D. Haussler (Ed.), Pittsburgh, PA, 1992, pp. 144-152.

[2] S. Abe, *Support Vector Machines for Pattern Classification*, London: Springer, 2005.

[3] R. Li, et al., "Coevolutionary feature synthesized EM algorithm for image retrieval," in *Proc. ACMMM*, Singapore, 2005, pp. 696-705.

[4] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature,* Vol. 401, pp. 788-791, 1999.

[5] S. Z. Li, et al., "Learning spatially localized, parts-based representation," in *Proc. CVPR*, 2001, pp. 207-212.

[6] W. Xu, et al., "Document clustering based on non-negative matrix factorization," in *Proc. SIGIR*, 2003, pp. 267-273.

[7] X. He, et al., "Learning an image manifold for retrieval," in *Proc. ACMMM*, New York, USA, 2004, pp. 17-23.

[8] X. He, et al., "Learning a locality preserving subspace for visual recognition," in *Proc. ICCV*, 2003, pp. 385-393.

[9] Y. Lin and B. Bhanu, "Evolutionary feature synthesis for object recognition," *IEEE Trans. on SMC, Part C,* Vol. 35, No. 2, pp. 156-171, 2005.

[10] A. Dong, et al., "Evolutionary feature synthesis for image databases," in *IEEE WACV*, 2005, pp. 330-335.

[11] A. P. Dempster, et al., "Maximum likelihood from incomplete data via the EM algorithm with discussion," *Journal of the Royal Statistical Society (Series B),* Vol. 39, No. 1, pp. 1-38, 1977.

[12] R. d. S. Virginia, "Learning classification with unlabeled data," in *Advances in Neural Information Processing Systems*: Morgan Kaufmann, 1993, pp. 112-119.

[13] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. ICML*, 1999.

[14] D. Bargeron, et al., "Boosting-based transductive learning for text detection," in *Proc. Int. Conf. on Document Analysis and Recognition*, 2005, Vol. II, pp. 1166-1171.

[15] Y. Wu and T. S. Huang, "Color tracking by transductive learning," in *Proc. CVPR*, 2000, pp. 133-138.

[16] Y. Wu, et al., "Discriminant-EM algorithm with application to image retrieval," in *Proc. CVPR*, 2000, pp. 222-227.

[17] C. Saunders, et al., "Transduction with confidence and credibility," in *Proc. IJCAI*, 1999, Vol. 2, pp. 722-726.

[18] B. Bhanu, et al., *Evolutionary synthesis of pattern recognition systems*, New York: Springer-Verlag, 2005.

[19] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, Cambridge MA: MIT Press, 1994.

[20] G. McLachlan and D. Peel, *Finite Mixture Models*, Wiley, 2000.

[21] A. Dong and B. Bhanu, "Active concept learning in image databases," *IEEE Trans. on SMC, Part B,* Vol. 35, No. 3, pp. 450-466, 2005.

[22] P. Y. Yin, et al., "Integrating relevance feedback techniques for image retrieval using reinforcement learning," *IEEE Trans. on PAMI,* Vol. 27, No. 10, pp. 1536-1551, 2005.

[23] X. S. Zhou, et al., "Water-filling: a novel way for image structural feature extraction," in *Proc. ICIP*, Kobe, Japan, 1999.

[24] C. C. Chang and C. J. Lin, "LIBSVM : a library for support vector machines," *Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm*, 2001.

[25] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods -- Support Vector Learning*, B. Schlkopf, et al., (Eds.), Cambridge, MA: MIT Press, 1999, pp. 169-184.